



Thyra from a Developer's Perspective

Roscoe A. Bartlett

Department 1411: Optimization and Uncertainty Estimation

Sandia National Laboratories

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy under contract DE-AC04-94AL85000.





Outline

- Quick overview of abstract numerical algorithms (ANAs), Trilinos software, interfaces, and computing environments
- Why interoperability and standard interfaces are important
- Requirements for Thyra and the Fundamental ANA Operator/Vector Interfaces
- Three different general categories of use cases for Thyra ANA interfaces and required vs. optional software and capability vs. dependency
- Thyra ANA operator/vector adapter support software
 - Current Epetra/Thyra adapters or lack of?
- Trilinos Thyra package structure



Outline

- Quick overview of abstract numerical algorithms (ANAs), Trilinos software, interfaces, and computing environments
- Why interoperability and standard interfaces are important
- Requirements for Thyra and the Fundamental ANA Operator/Vector Interfaces
- Three different general categories of use cases for Thyra ANA interfaces and required vs. optional software and capability vs. dependency
- Thyra ANA operator/vector adapter support software
 - Current Epetra/Thyra adapters or lack of?
- Trilinos Thyra package structure



Categories of Abstract Problems and Abstract Algorithms

Trilinos Packages

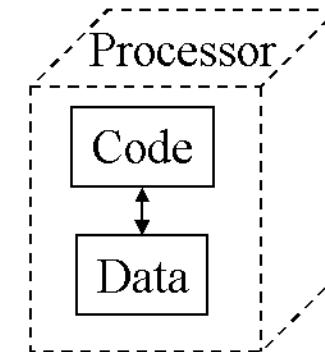
- Linear Problems: Given linear operator (matrix) $A \in \mathbf{R}^{n \times n}$
 - Linear equations: Solve $Ax = b$ for $x \in \mathbf{R}^n$ Belos
 - Eigen problems: Solve $Av = \lambda v$ for (all) $v \in \mathbf{R}^n$ and $\lambda \in \mathbf{R}$ Anasazi
- Nonlinear Problems: Given nonlinear operator $c(x, u) \in \mathbf{R}^{n+m} \rightarrow \mathbf{R}^n$
 - Nonlinear equations: Solve $c(x) = 0$ for $x \in \mathbf{R}^n$ NOX
 - Stability analysis: For $c(x, u) = 0$ find space $u \in \mathcal{U}$ such that $\frac{\partial c}{\partial x}$ is singular LOCA
- Transient Nonlinear Problems:
 - DAEs/ODEs: Solve $f(\dot{x}(t), x(t), t) = 0, t \in [0, T], x(0) = x_0, \dot{x}(0) = x'_0$ for $x(t) \in \mathbf{R}^n, t \in [0, T]$ Rythoms
- Optimization Problems:
 - Unconstrained: Find $u \in \mathbf{R}^n$ that minimizes $f(u)$
 - Constrained: Find $y \in \mathbf{R}^m$ and $u \in \mathbf{R}^n$ that:
minimizes $f(y, u)$
such that $c(y, u) = 0$ MOOCHO



Common Environments for Scientific Computing

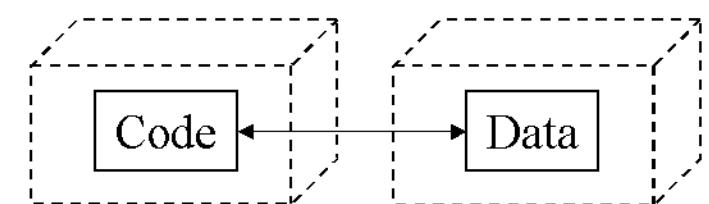
Serial / SMP (symmetric multi-processor)

- All data stored in RAM in a single local process



Out of Core

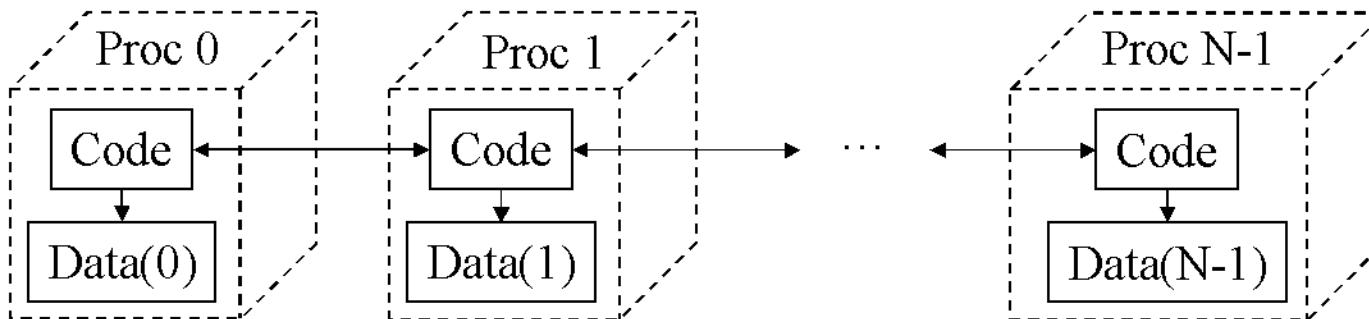
- Data stored in file(s) (too big to fit in RAM)



SPMD (Single program multiple data)

- Same code on each processor but different data

MPP (massively parallel processors)





Introducing Abstract Numerical Algorithms

What is an abstract numerical algorithm (ANA)?

An ANA is a numerical algorithm that can be expressed abstractly solely in terms of vectors, vector spaces, and linear operators (i.e. not with direct element access)

Example Linear ANA (LANA) : Linear Conjugate Gradients

Given:

$A \in \mathcal{X} \rightarrow \mathcal{X}$: s.p.d. linear operator

$b \in \mathcal{X}$: right hand side vector

Find vector $x \in \mathcal{X}$ that solves $Ax = b$

Linear Conjugate Gradient Algorithm

Compute $r^{(0)} = b - Ax^{(0)}$ for the initial guess $x^{(0)}$.

for $i = 1, 2, \dots$

$$\rho_{i-1} = \langle r^{(i-1)}, r^{(i-1)} \rangle$$

$$\beta_{i-1} = \rho_{i-1}/\rho_{i-2} (\beta_0 = 0)$$

$$p^{(i)} = r^{(i-1)} + \beta_{i-1} p^{(i-1)} (p^{(1)} = r^{(1)})$$

$$q^{(i)} = Ap^{(i)}$$

$$\gamma_i = \langle p^{(i)}, q^{(i)} \rangle$$

$$\alpha_i = \rho_{i-1}/\gamma_i$$

$$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$$

$$r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$$

check convergence; continue if necessary

end

Types of operations

linear operator applications

Linear Operators

- A

vector-vector operations

Vectors

- r, x, p, q

Scalar operations

Scalars

- $\rho, \beta, \gamma, \alpha$

scalar product
 $\langle x, y \rangle$ defined by
vector space

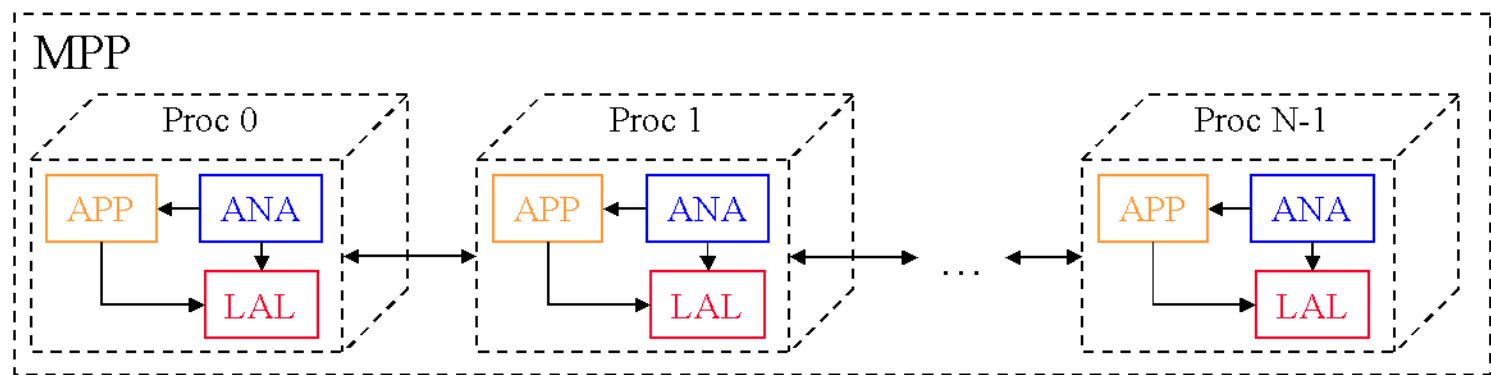
Vector spaces?

- \mathcal{X}

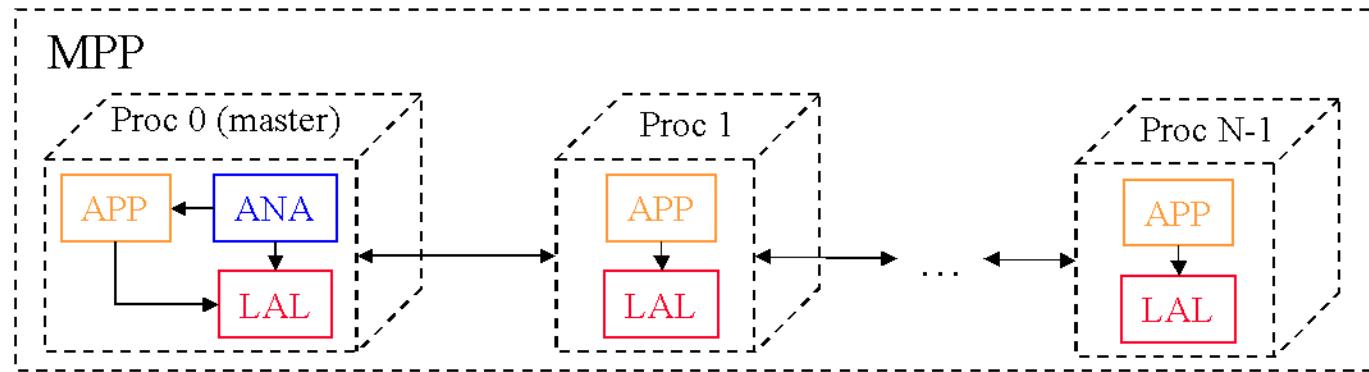


Different Platform Configurations for Running ANAs

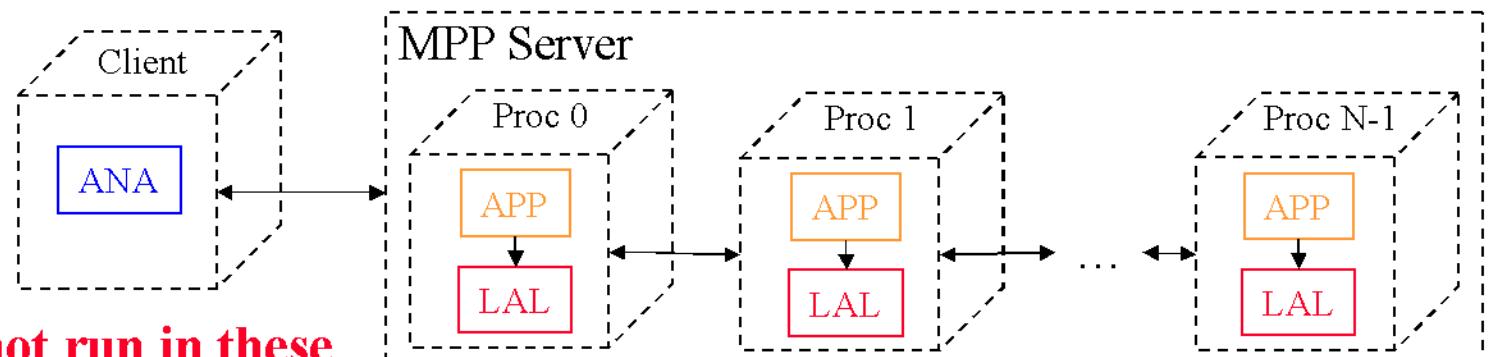
SPMD



Master/Slave



**Client/Server
Master/Slave**

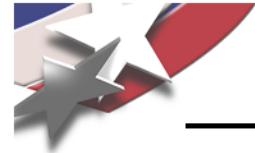


If a code can not run in these modes it is not an ANA code!



Outline

- Quick overview of abstract numerical algorithms (ANAs), Trilinos software, interfaces, and computing environments
- Why interoperability and standard interfaces are important
- Requirements for Thyra and the Fundamental ANA Operator/Vector Interfaces
- Three different general categories of use cases for Thyra ANA interfaces and required vs. optional software and capability vs. dependency
- Thyra ANA operator/vector adapter support software
 - Current Epetra/Thyra adapters or lack of?
- Trilinos Thyra package structure



Trilinos Strategic Goals

- **Scalable Solvers:** As problem size and processor counts increase, the cost of the solver will remain a nearly fixed percentage of the total solution time.
- **Hardened Solvers:** Never fail unless problem essentially unsolvable, in which case we diagnose and inform the user why the problem fails and provide a reliable measure of error.
- **Full Vertical Coverage:** Provide leading edge capabilities from basic linear algebra to transient and optimization solvers.
- **Universal Interoperability:** All Trilinos packages will be interoperable, so that any combination of solver packages that makes sense algorithmically will be possible within Trilinos.
- **Universal Solver RAS:** Trilinos will be:
 - Integrated into every major application at Sandia (Availability).
 - The leading edge hardened, scalable solutions for each of these applications (Reliability).
 - Easy to maintain and upgrade within the application environment (Serviceability).

Thyra is being developed to address this issue

Courtesy of Mike Heroux, Trilinos Project Leader

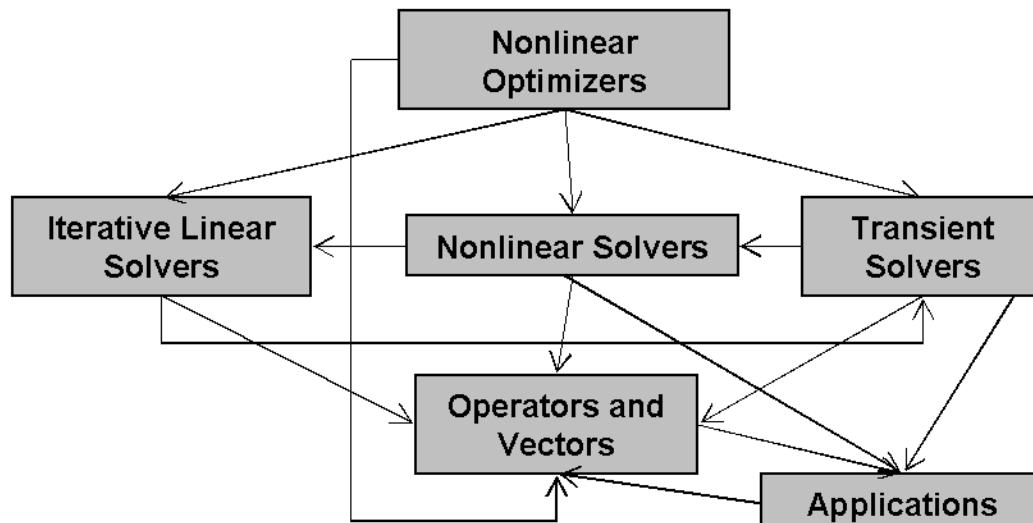
Key Point

- Universal Interoperability will not happen automatically and if not done carefully then can compromise the other strategic goals



Interoperability is Especially Important to Optimization

Numerous interactions exist between abstract numerical algorithms (ANAs) in a transient optimization problem



What is needed to solve problem?

- Standard interfaces to break $O(N^2)$
- 1-to-1 couplings
 - Operators/vectors
 - Linear Solvers
 - Nonlinear solvers
 - Transient solvers
 - etc.

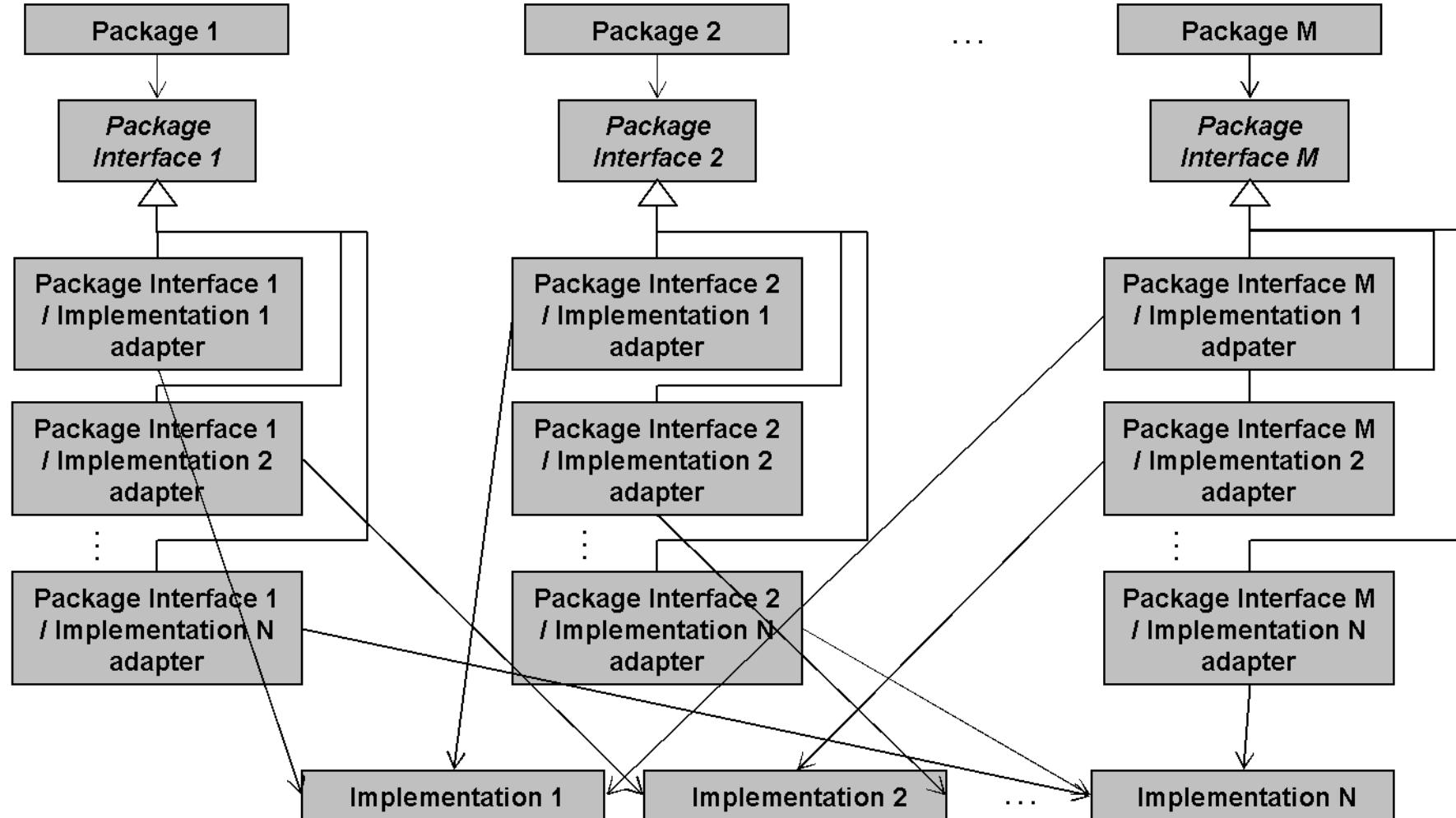
This is hard! This level of interoperability for massively parallel algorithms has never been achieved before!

Key Points

- Higher level algorithms, like optimization, require a lot of interoperability
- Interoperability must be “easy” or these configurations will not be possible
- Many real problems even more complicated



Interfacing Packages to Implementations : Everyone for themselves!

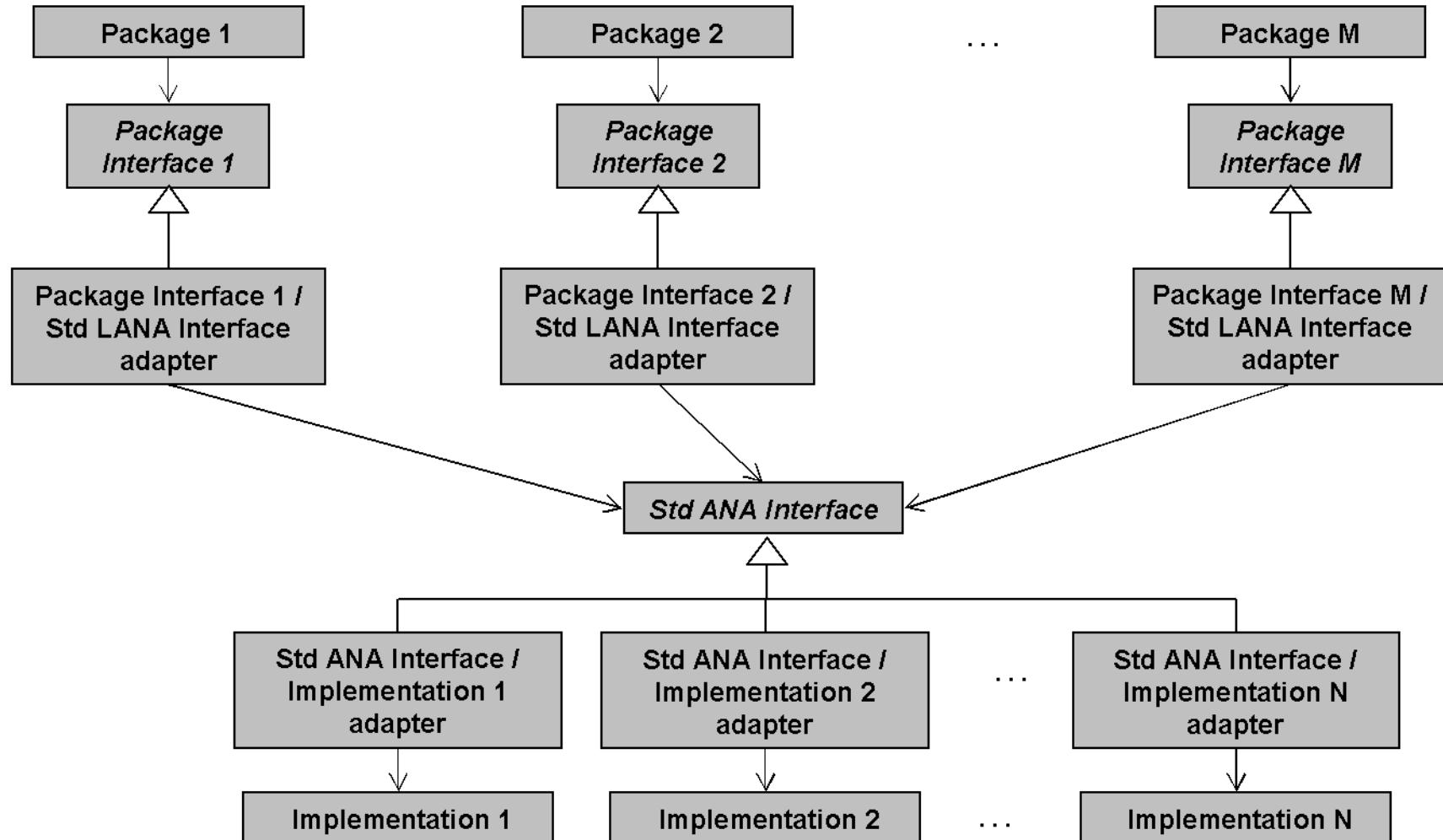


- Oh no, $M * N$ adapter subclasses needed!
- This is not a “scalable” approach!

Package == ANA



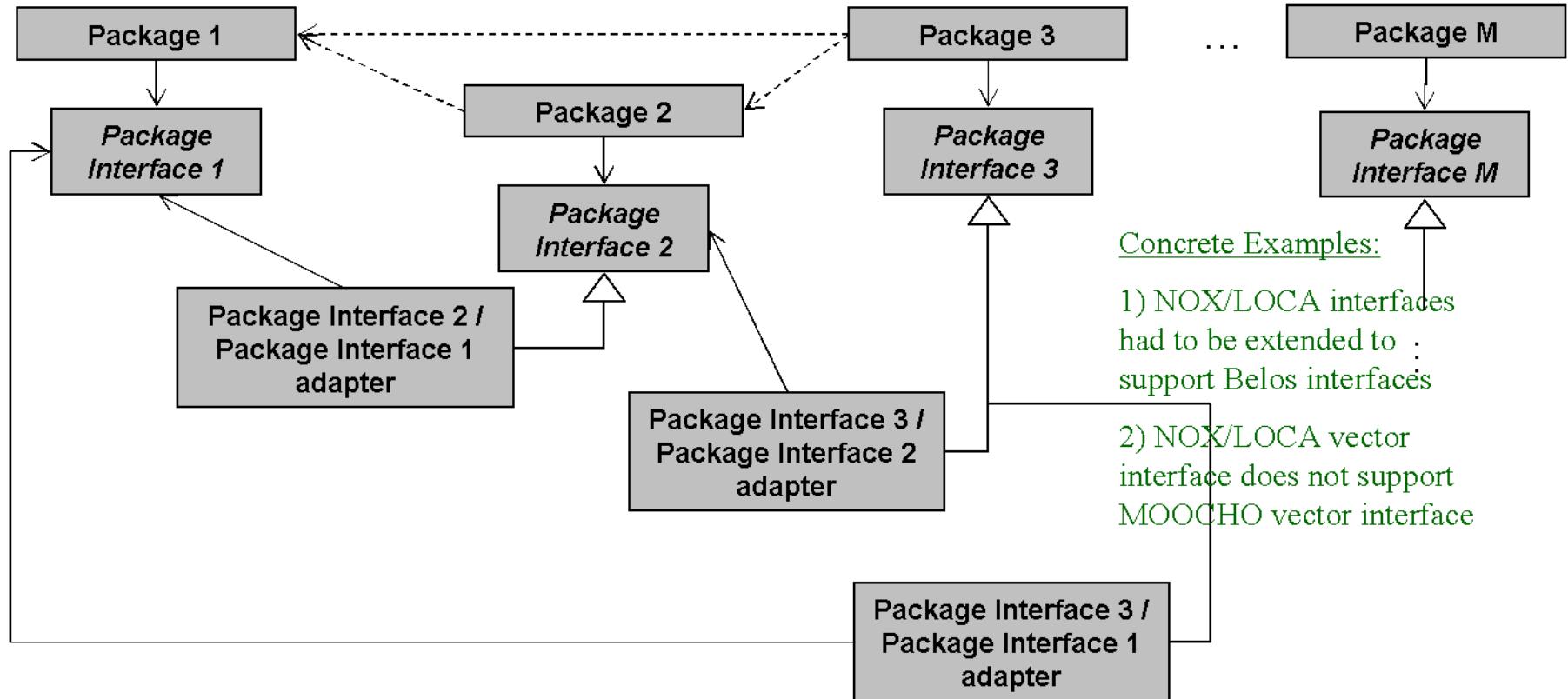
Interfacing Packages to Implementations : Standard Interfaces



- + Only $M + N$ adapter subclasses needed!
- + This is a “scalable” approach!



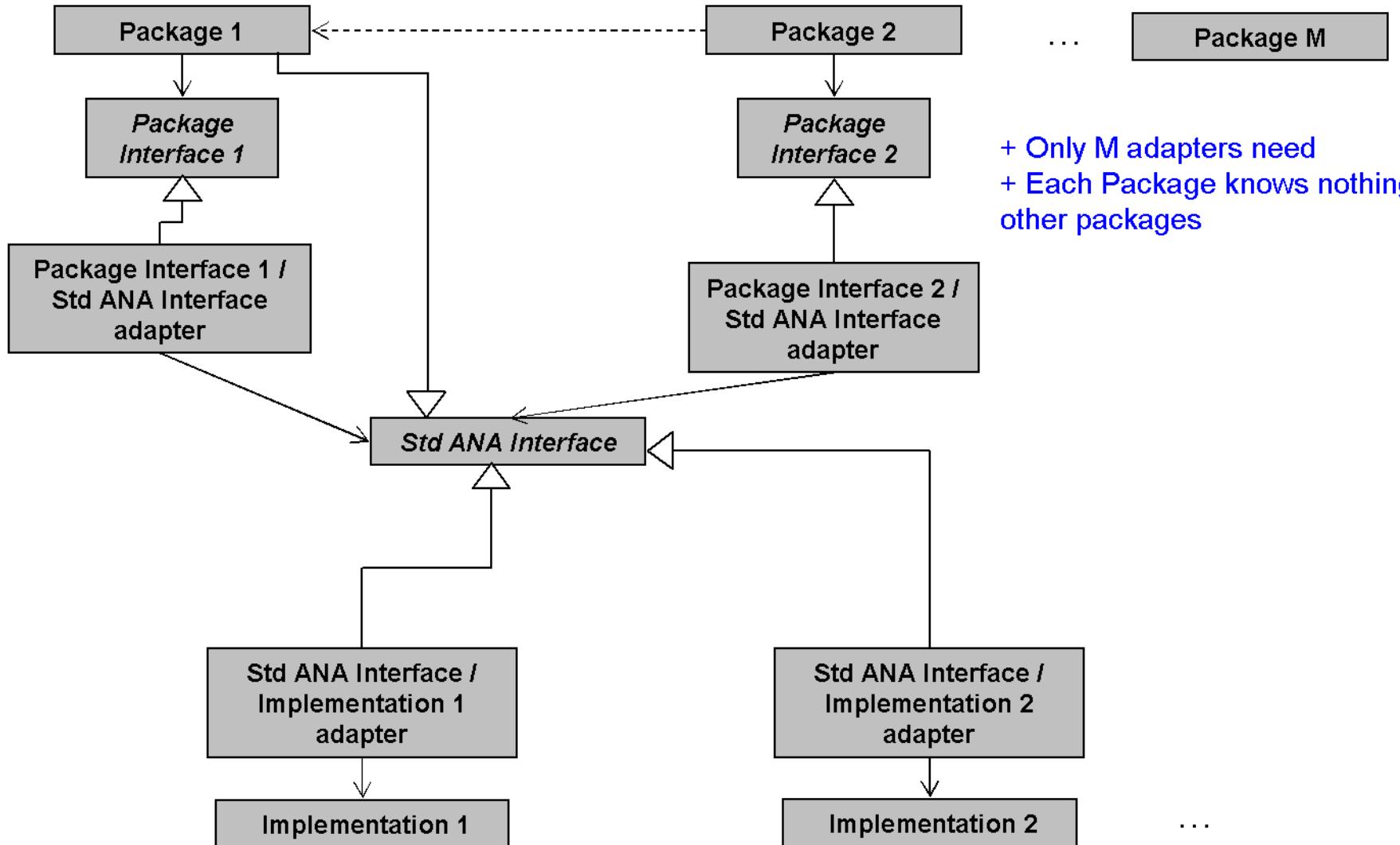
Interfacing Packages to Packages : Everyone for themselves!



- Major assumption: “Package Interface 1” satisfies the requirements of “Package Interface 2”?
- Oh no, as many as $1 + 2 + 3 + \dots + M-1 = O(M^2)$ adapter subclasses needed if only one-way interactions!
- This is not a “scalable” approach!



Interfacing Packages to Packages : Standard Interfaces



The Recommended Practice!



Reasons to Adopt Standard Interfaces

- “Automatic” interoperability only comes through “standard” interfaces
- Only way to guarantee interoperability is even possible (without revisions)
- One set of common documentation
- Common set of unit tests for common interface objects
- More uniform access to Trilinos packages for users



Outline

- Quick overview of abstract numerical algorithms (ANAs), Trilinos software, interfaces, and computing environments
- Why interoperability and standard interfaces are important
- Requirements for Thyra and the Fundamental ANA Operator/Vector Interfaces
- Three different general categories of use cases for Thyra ANA interfaces and required vs. optional software and capability vs. dependency
- Thyra ANA operator/vector adapter support software
 - Current Epetra/Thyra adapters or lack of?
- Trilinos Thyra package structure



Requirements for Abstract Numerical Algorithms and Thyra

An important consideration \Rightarrow Scientific computing is computationally expensive!

Abstract Interfaces for Abstract Numerical Algorithms using Thyra must:

- Be portable to all ASC (advanced scientific computing) computer platforms
- Provide for stable and accurate numerics
- Result in algorithms with near-optimal storage and runtime performance
 - Scientific computing is expensive!

An important ideal \Rightarrow A customized hand-code algorithm in Fortran 77 should not provide significant improvements in storage requirements, speed or numerical stability!

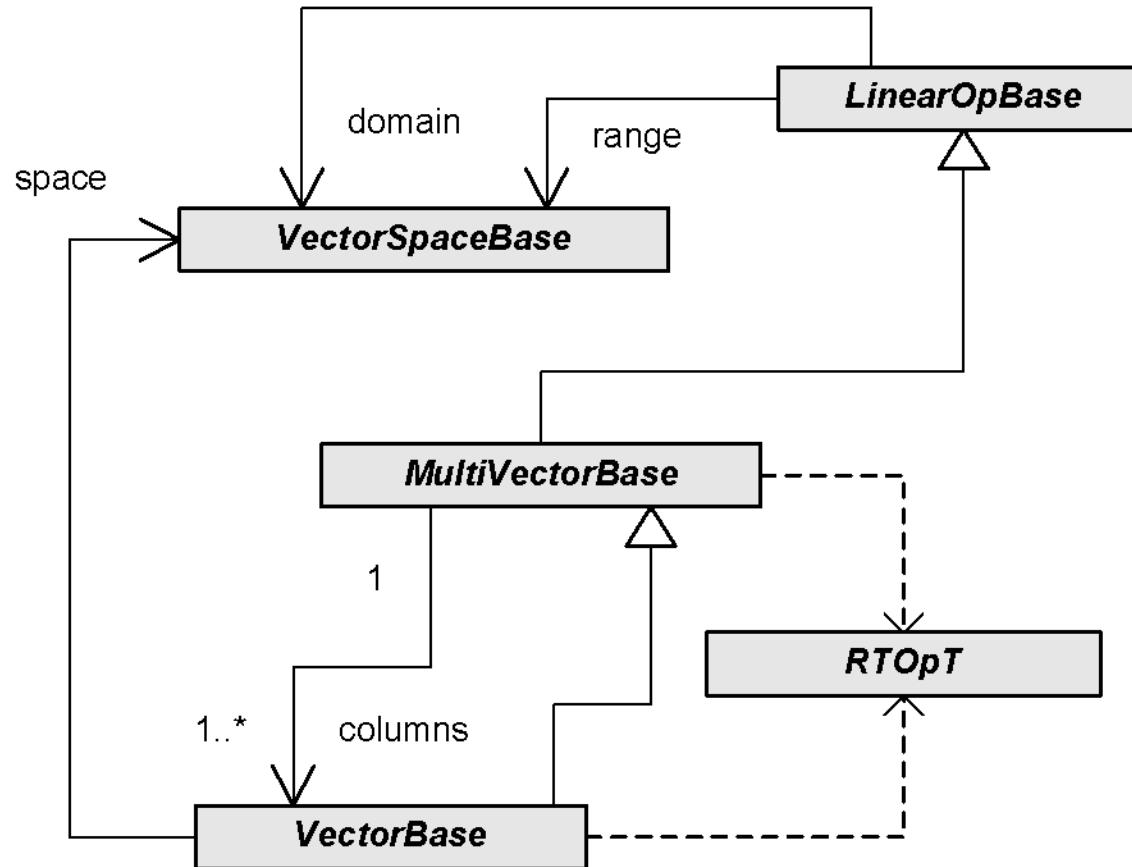
An important ideal \Rightarrow Object-oriented “overhead” should be constant and not increase as the problem size increases.

Abstract Interfaces for Abstract Numerical Algorithms using Thyra should:

- Be minimal but complete (good object-oriented design principle)
- Support a variety of computing platforms and configurations
 - i.e. Serial/SMP, Out-of-Core, SPMD, master/slave and client/server
- Be (relatively) easy to provide new implementations (**Subjective!!!**)
- Not be too complicated (**Subjective!!!**)



Foundational Thyra ANA Operator/Vector Interfaces



The Key to success!
Reduction/Transformation Operators

- Supports all needed vector operations
- Data/parallel independence
- Optimal performance

R. A. Bartlett, B. G. van Bloemen Waanders and M. A. Heroux. Vector Reduction/Transformation Operators, ACM TOMS, March 2004

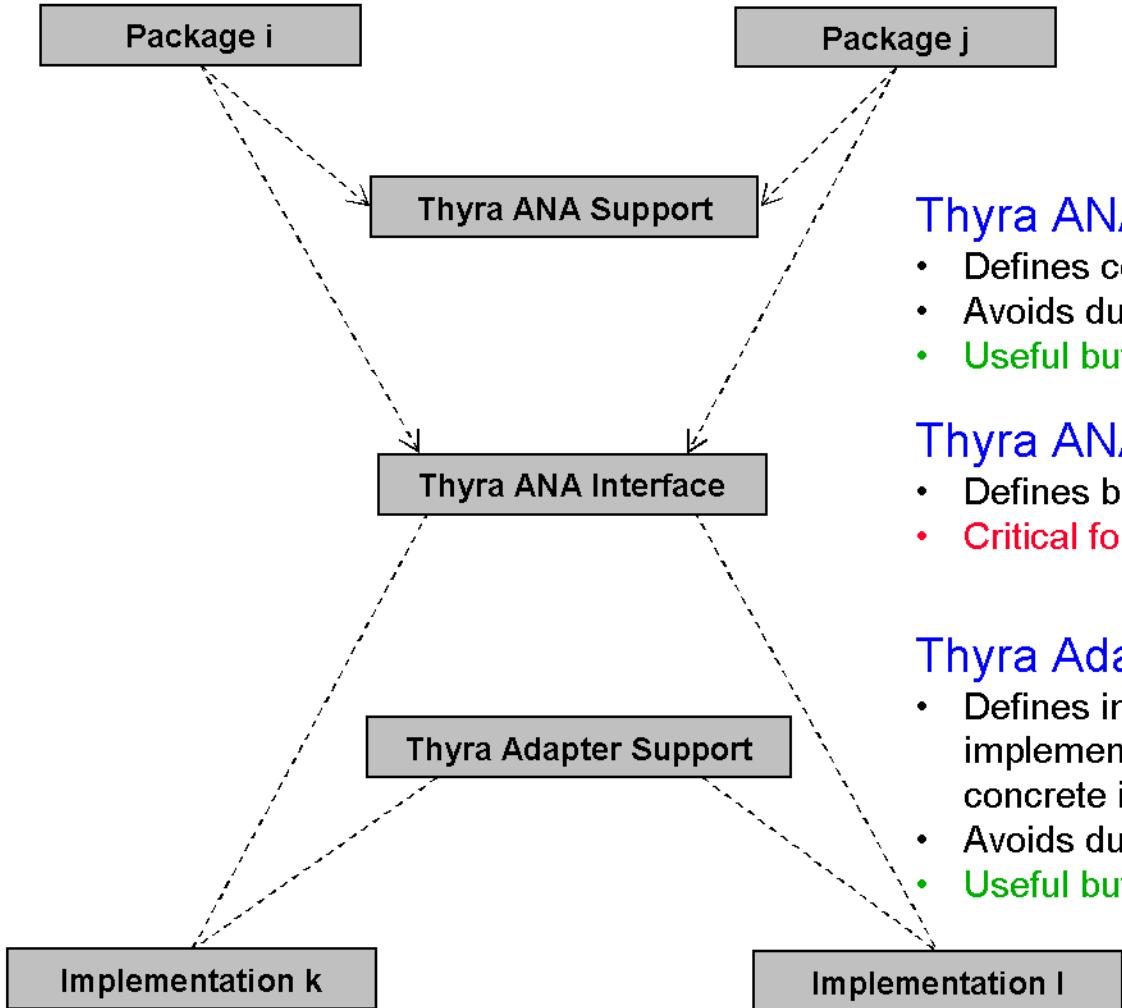


Outline

- Quick overview of abstract numerical algorithms (ANAs), Trilinos software, interfaces, and computing environments
- Why interoperability and standard interfaces are important
- Requirements for Thyra and the Fundamental ANA Operator/Vector Interfaces
- Three different general categories of use cases for Thyra ANA interfaces and required vs. optional software and capability vs. dependency
- Thyra ANA operator/vector adapter support software
 - Current Epetra/Thyra adapters or lack of?
- Trilinos Thyra package structure



Three Different Types of Use Cases for Thyra ANA Interfaces



Thyra ANA Support Software

- Defines conveniences to aid in writing ANAs
- Avoids duplication of effort
- **Useful but optional!**

Thyra ANA Interoperability Interfaces

- Defines basic functionality needed for ANAs
- **Critical for scalable interoperability!**

Thyra Adapters Support Software

- Defines infrastructure support and concrete implementations to make it easy to provide concrete implementations for Thyra ANA interfaces
- Avoids duplication of effort
- **Useful but optional!**

The question should not be if this ANA support and adapter support software should exist but instead the question should be where this software should be placed and what relationship it will have to the Thyra ANA interoperability interfaces



Outline

- Quick overview of abstract numerical algorithms (ANAs), Trilinos software, interfaces, and computing environments
- Why interoperability and standard interfaces are important
- Requirements for Thyra and the Fundamental ANA Operator/Vector Interfaces
- Three different general categories of use cases for Thyra ANA interfaces and required vs. optional software and capability vs. dependency
- [Thyra ANA operator/vector adapter support software](#)
 - Current Epetra/Thyra adapters or lack of?
- Trilinos Thyra package structure



Thyra Adapter Support for APP-specific Scalar Products

Linear operators must “obey” application-specific scalar product $\langle x, y \rangle$

- Specifically, for the linear operator (i.e. LinearOpBase, MultiVectorBase etc.):

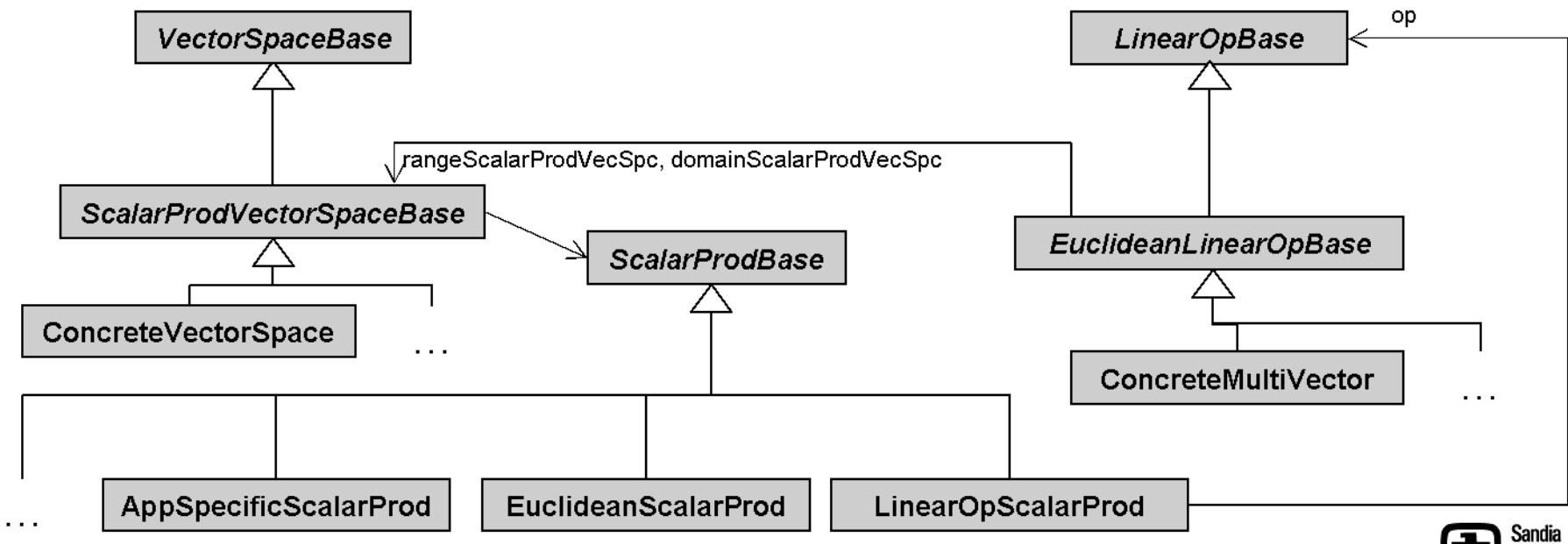
$$A \in \mathcal{D} \rightarrow \mathcal{R}$$

- the following adjoint relation must hold:

$$\langle u, Av \rangle_{\mathcal{R}} = \langle A^H u, v \rangle_{\mathcal{D}}$$

Goal of Thyra adapters support subclasses

=> Separate the definition of application-specific scalar product from data-structure and computing platform concrete implementations of vector spaces, vectors, multi-vectors and linear operators as much as possible.



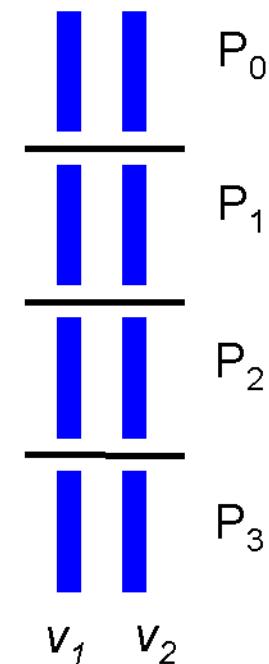


Automatic Compatibility of Serial and SPMD (Multi)Vectors

A simple principle for compatibility of (Multi)Vectors

As set of vector (or multi-vector) objects v_1, v_2, \dots, v_n should be automatically and efficiently compatible, regardless of their concrete implementation, if their coefficients are co-located in the same address space, period!

- **Example:** If two vectors v_1 and v_2 in an SPMD program have their elements partitioned to processors in the same way then these two concrete implementations should be interchangeable for any client software that fills/accesses these vectors.
- **Anti-example:** TNT vectors are not compatible with any (nontemplated) software that accepts `std::vector` representations.
- This type of compatibility for serial and SPMD vectors and multi-vectors is automatically supported as built in to the basic Thyra interfaces `VectorBase` and `MultiVectorBase`!





Thyra Interface Support for (Multi)Vector Compatibility

- Automatic (Multi)Vector object compatibility is supported through explicit coefficient views:

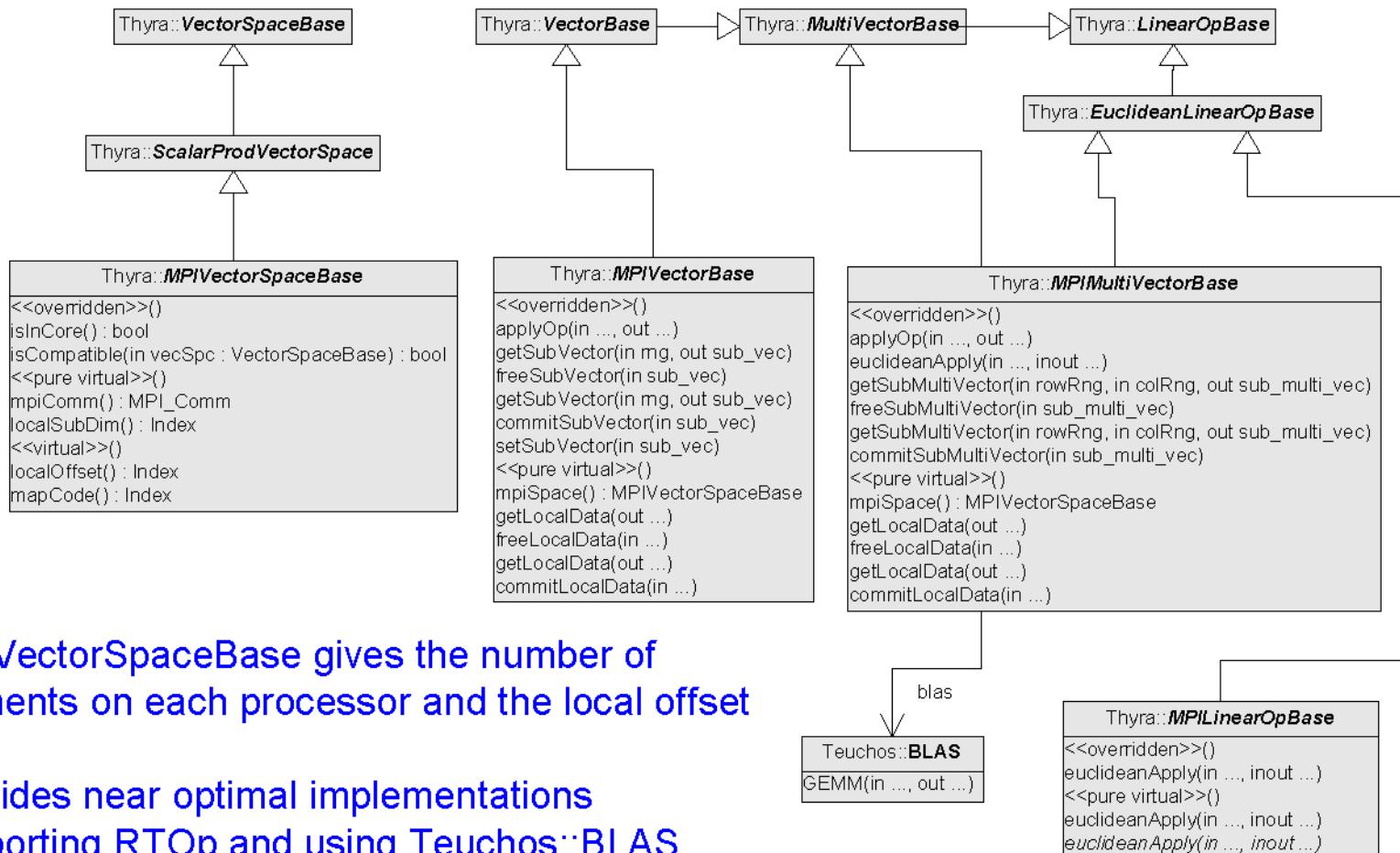
```
template<class Scalar>
class VectorBase : virtual public MultiVectorBase<Scalar> {
public:
    ...
    virtual void getSubVector( const Rang1D& rng, RTOpPack::SubVectorT<Scalar>* sub_vec ) const;
    virtual void freeSubVector( RTOpPack::SubVectorT<Scalar>* sub_vec ) const;
    virtual void getSubVector( const Rang1D& rng, RTOpPack::MutableSubVectorT<Scalar>* sub_vec );
    virtual void commitSubVector( RTOpPack::MutableSubVectorT<Scalar>* sub_vec );
    ...
};

template<class Scalar>
class MultiVectorBase : virtual public LinearOpBase<Scalar> {
public:
    ...
    virtual void getSubMultiVector(
        const Rang1D &rowRng, const Rang1D &colRng
        ,RTOpPack::SubMultiVectorT<Scalar> *sub_mv ) const;
    virtual void freeSubMultiVector( RTOpPack::SubMultiVectorT<Scalar>* sub_mv ) const;
    virtual void getSubMultiVector(
        const Rang1D &rowRng, const Rang1D &colRng
        ,RTOpPack::MutableSubMultiVectorT<Scalar> *sub_mv );
    virtual void commitSubMultiVector( RTOpPack::MutableSubMultiVectorT<Scalar>* sub_mv );
    ...
};
```

- The types Sub(Multi)VectorT and MutableSub(Multi)VectorT simply store raw pointers to memory and information about where they came from the host (Multi)Vector.
- Consequence: If you know the partitioning of elements to processors then you can access the coefficients of any (Multi)Vector object without any dynamic casting!



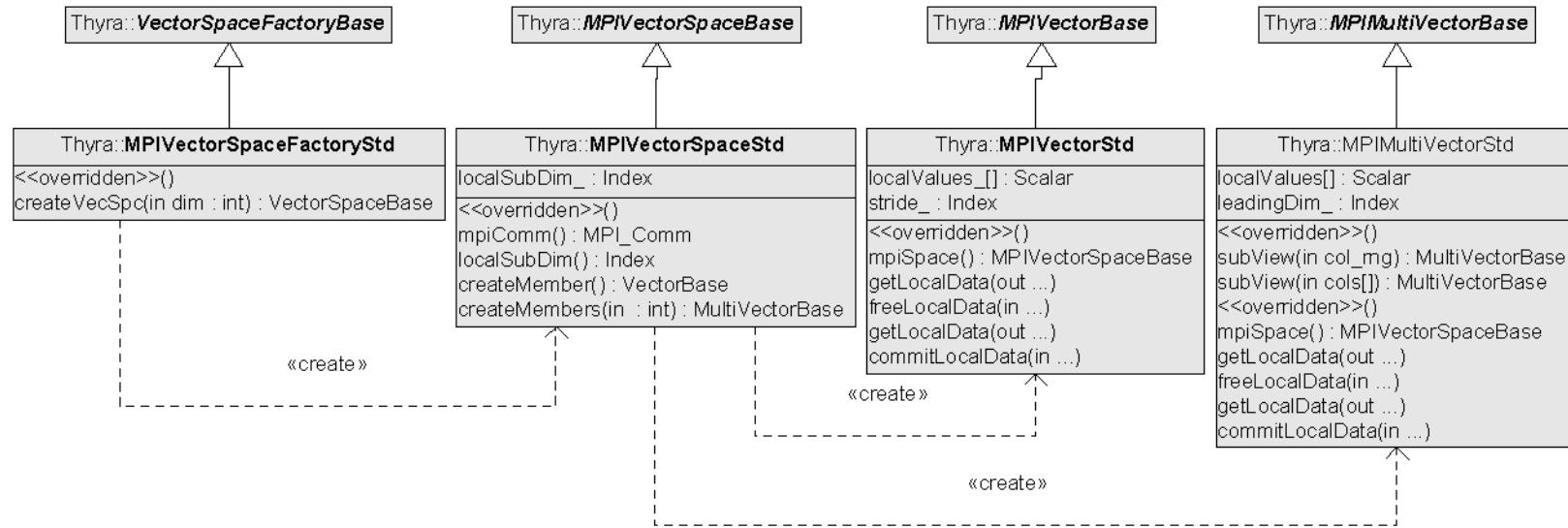
MPI-based SPMD Operator/Vector Node Subclasses



- **MPIVectorSpaceBase** gives the number of elements on each processor and the local offset
- Provides near optimal implementations supporting RTOp and using Teuchos::BLAS
- Concrete subclasses must just provide contiguous views of data on a processor
- (Multi)Vectors are automatically compatible if their (range) space object support the MPIVectorSpaceBase interface! => No dynamic casting of (Multi)Vector objects!
- **MPILinearOpBase** provides explicit access to (multi)vector elements and takes care of the range and domain spaces.



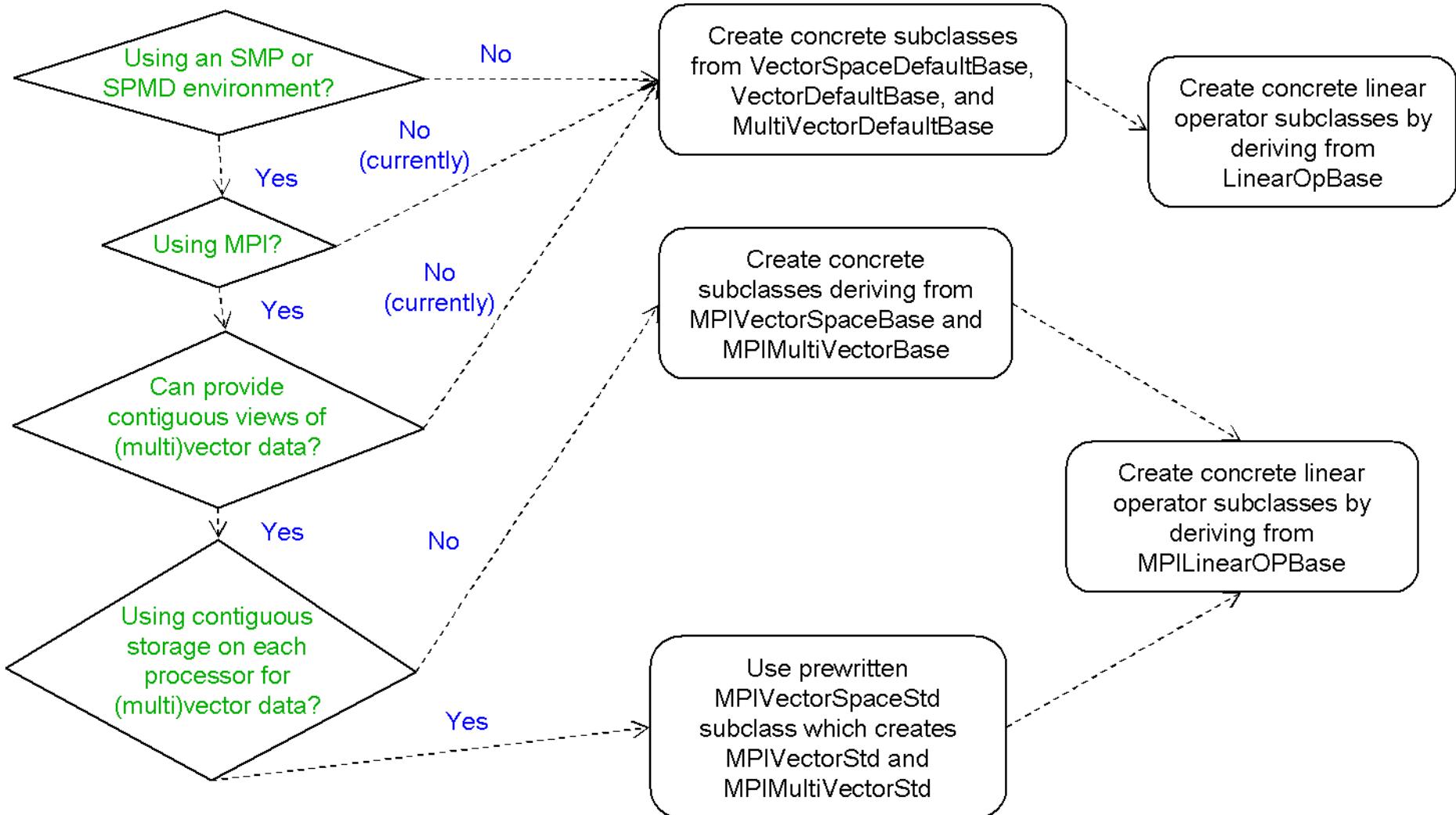
Concrete MPI-based SPMD Operator/Vector Subclasses



- Concrete Vector and MultiVector subclasses accept `RefCountPtr`-wrapped pointers to raw array data or can construct the arrays internally
- `MPIVectorStd` and `MPIMultiVectorStd` can be used as base classes when explicit contiguous storage is used by implementation.
- `MPIMultiVectorStd` provides highly efficient implementations of the contiguous and non-contiguous `subView()` functions.
- These concrete subclasses provide highly efficient, (near) optimal implementations of vectors and multi-vectors for most use cases.



Steps to Implementing Concrete Thyra ANA Objects



If: (a) Using SPMD and MPI for communication, (b) All (multi)vector data is uniquely stored in contiguous arrays on each processor

Then: the `xxxStd` subclasses should be near optimal for most if not all use cases!



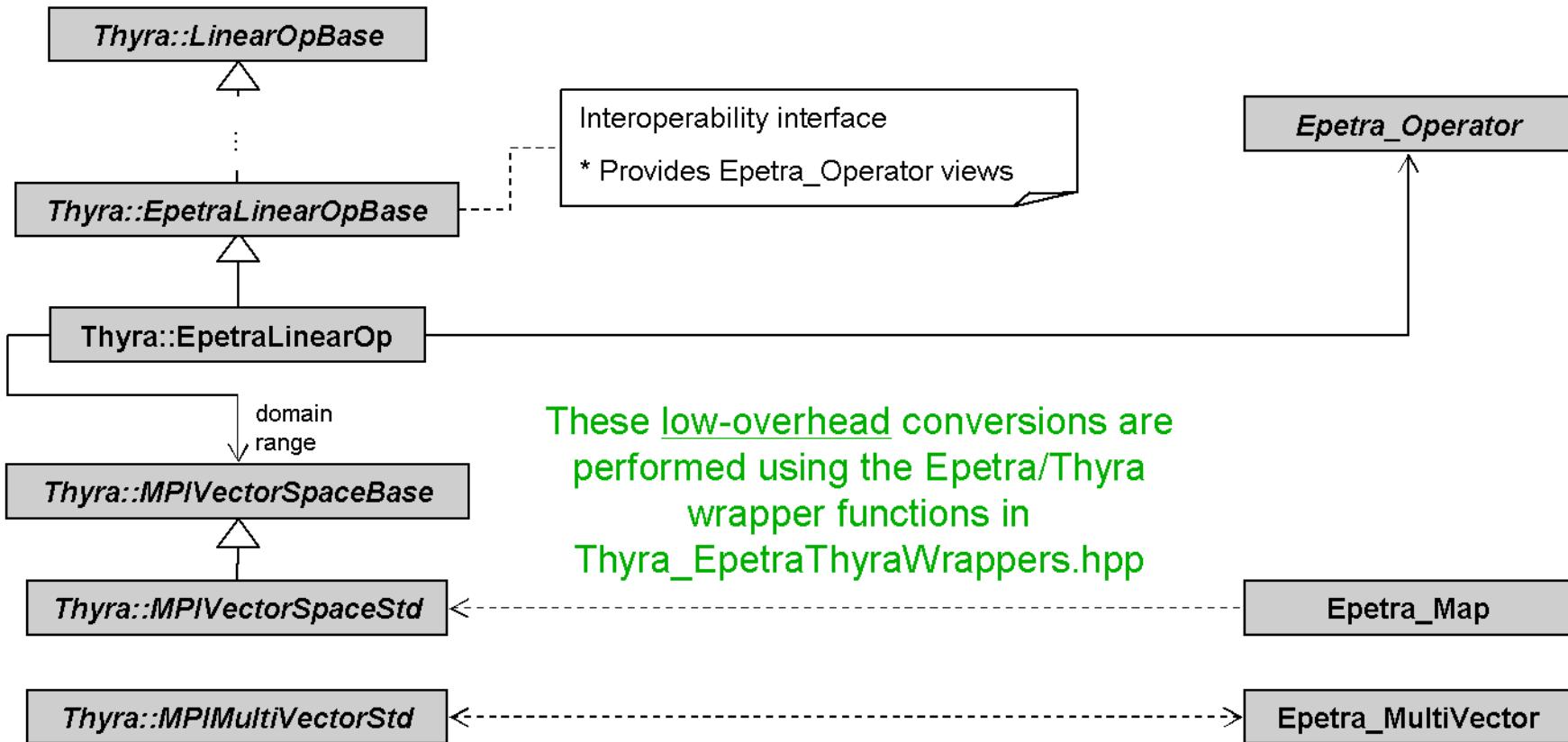
Outline

- Quick overview of abstract numerical algorithms (ANAs), Trilinos software, interfaces, and computing environments
- Why interoperability and standard interfaces are important
- Requirements for Thyra and the Fundamental ANA Operator/Vector Interfaces
- Three different general categories of use cases for Thyra ANA interfaces and required vs. optional software and capability vs. dependency
- Thyra ANA operator/vector adapter support software
 - Current Epetra/Thyra adapters or lack of?
- Trilinos Thyra package structure



Epetra/Thyra Operator/Vector Adapters

Current Epetra/Thyra adapters are a mix of “real” adapters and “fake” adapters



These low-overhead conversions are performed using the Epetra/Thyra wrapper functions in `Thyra_EpetraThyraWrappers.hpp`

This style of adapters:

- Allows for **automatic compatibility** of all *similar* (Multi)Vector object implementations
- Requires **less code** to maintain than the “real” adapters
- Provides **greater opportunities** for optimization of ANA-only operations
- **Can be changed** without affecting clients by using wrapper functions
- Outstanding issues: **Flop counts?** Others ???



Performance of Epetra/Thyra Adapters vs. Pure Epetra

Test program put together by Chris Baker for Anasazi

1) Apply Operator:

$$V = A * X$$

where: X and V
are numel x n

2) Block multivector update
(MvTimeMatAddMv)

$$Z = \text{alpha} * Z + X * T$$

where: Z is numel x n, X is numel x m,
and T is m x n

3) Block inner (i.e. dot) product
(MvTimesMv)

$$T = X * Y$$

where: X is numel x m,
Y is numel x n,
and T is m x n

In these tests numel=50000, m=100,
and n=5.

Times originally reported by Chris Baker

	epetra	thyra	thyra/epetra
FULL: Apply Operator	: 0.2445893	0.2413893	0.2429688
FULL: MvTimeMatAddMv	: 0.2813016	0.2760603	0.2809743
FULL: MvTransMv	: 0.1958322	0.189001	0.1878586
CONT-VIEW: Apply Operator	: 0.2400045	0.238128	0.2429735
CONT-VIEW: MvTimeMatAddMv	: 0.2892126	0.3099845	0.3125794
CONT-VIEW: MvTransMv	: 0.1893584	0.1986764	0.2006157
VIEW: Apply Operator	: 0.3461629	0.7728371	0.7761013
VIEW: MvTimeMatAddMv	: 0.4013208	0.2982232	0.3071349
VIEW: MvTransMv	: 0.3055691	0.3063636	0.305717

Similar performance for whole multi-vectors
and contiguous multi-vector views

Both horrible and decent performance for
non-contiguous multi-vector views (i.e. first
and last columns interchanged)

After minor modification to Thyra::SerialMultiVectorStd and Thyra::MPIMultiVectorStd

	epetra	thyra	thyra/epetra
FULL: Apply Operator	: 0.153514	0.149969	0.14886
FULL: MvTimeMatAddMv	: 0.278658	0.27842	0.278105
FULL: MvTransMv	: 0.185737	0.185776	0.185522
CONT-VIEW: Creation	: 1.3e-05	3.9e-05	3.2e-05
CONT-VIEW: Apply Operator	: 0.151971	0.148705	0.14631
CONT-VIEW: MvTimeMatAddMv	: 0.278541	0.278497	0.27983
CONT-VIEW: MvTransMv	: 0.1857	0.185799	0.1864
VIEW: Creation	: 1.5e-05	0.116395	0.116881
VIEW: Apply Operator	: 0.328524	0.147952	0.146495
VIEW: MvTimeMatAddMv	: 0.38062	0.281685	0.279953
VIEW: MvTransMv	: 0.28367	0.185861	0.185714

Key Point
Dedicated yet general ANA concrete
implementations can out perform even
heavily used implementations like Epetra

Overall, better performance
for Epetra/Thyra adapters!

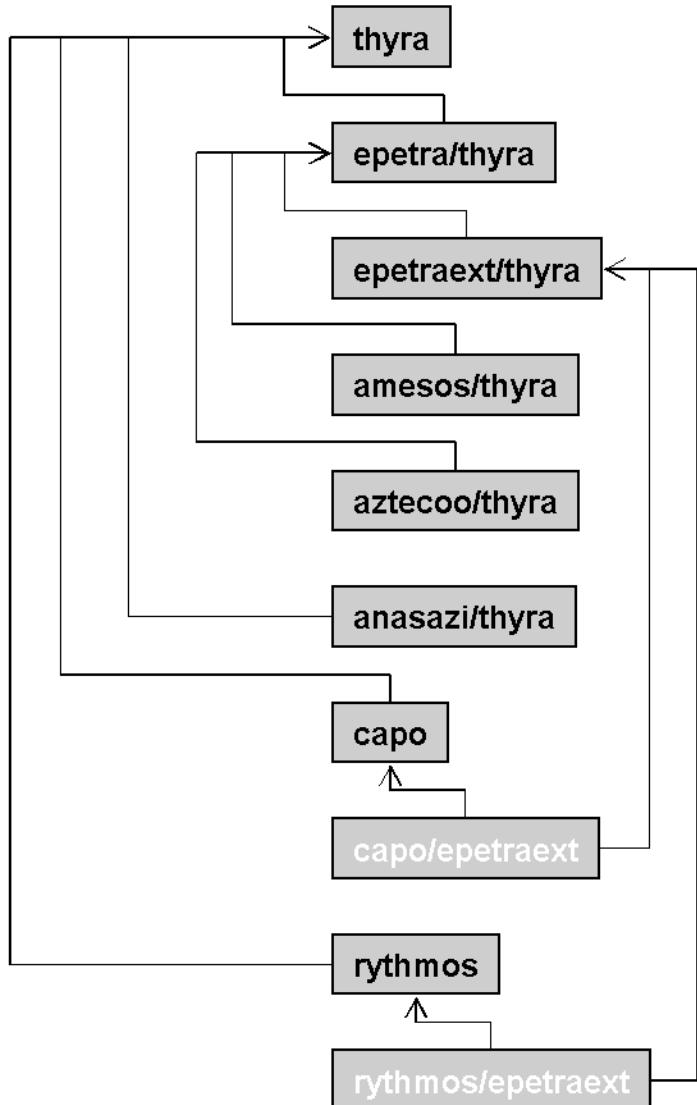


Outline

- Quick overview of abstract numerical algorithms (ANAs), Trilinos software, interfaces, and computing environments
- Why interoperability and standard interfaces are important
- Requirements for Thyra and the Fundamental ANA Operator/Vector Interfaces
- Three different general categories of use cases for Thyra ANA interfaces and required vs. optional software and capability vs. dependency
- Thyra ANA operator/vector adapter support software
 - Current Epetra/Thyra adapters or lack of?
- Trilinos Thyra package structure



Trilinos Package Structure Related to Thyra



- The ‘thyra’ package appears early in the build process
- Thyra adapters are placed in the packages where the concrete implementations reside
 - Allows for scalable growth in the number of Thyra adapters
 - Encourages native package developers to take ownership of their Thyra adapters
(Thanks Ansasizi developers!)
- CAPO and Rythmos are written directly in terms of Thyra interfaces!
- **Guidance:** Ask me or another Thyra developer to review your Thyra adapters



Summary

- **Thyra** Interfaces provide minimal but efficient connectivity between ANAs and linear algebra implementations and applications
- **Thyra** is the **critical** standard for interoperability between ANAs in Trilinos
- **Thyra** can be used in Serial/SMP, SPMD, client/server and master/slave
- **Thyra** provides a growing set of **optional** utilities for ANA development and subclass implementation support
- **Thyra** adapters are available for Epetra, Amesos, AztecOO, Anasazi, and Rythmos with others on the way (e.g. Belos, NOX, MOOCHO ...)

Trilinos website

<http://software.sandia.gov/trilinos>



The Principle of Control



- The ANA support and adapter support software in Thyra is like the machines that support Zion:

Key Point

- This software is for our convenience but we can throw it away if we want to and not lose any interoperability!